Machine Learning Report

Shengyuan Huang 516030910500 Yufan Cai 516030910496 Yingzhe Luo 516030910517

2019-06-06

1 Objective

我们的目标是解决一个关于细胞的多标签分类问题。

2 Contribution

黄晟原：

- 对数据进行预处理
- 做了数据增强
- 自定义了损失函数
- 训练了一个基础的CNN网络
- 重新训练了一个resnet50的网络

蔡雨凡：

- 对数据进行分析和预处理
- 使用VGG16模型做预测，测试使用数据增强、自定义损失函数的效果
- 使用划分和层次思想构建新模型

罗英哲：

- 使用预训练的模型Resnet50网络进行特征提取
- 运用Resnet构建端到端分类器

3 Part of Shengyuan Huang

3.1 Preprocess

首先我发现图片的大小不一，所以首先我想到的是将所有图片resize成同一大小，然后以numpy的形式保存下来以便于接下来的训练，从理论上来说，resize之后的图片越小，损失的信息越多，但是由于设备限制（内存不足，训练缓慢），我们无法将其resize成太大的图片。接着我读入图片标签的那个文件，并以字典的形式保存{图片名：标签}的键值对，每读入一张图，我就在字典里寻找其标签，并将其转化为一个长度为29 的numpy，每个位置的值为0 或1,1 表示属于这一类，0 表示不属于这一类。接着，我对要输入模型的numpy数组进行归一化以方便训练，由于rgb的值都在0-255 之间，所以我直接把要输入模型的numpy 数组除以255.0 就可以了，因为这样一来它的每个数都变到了0-1 之间。最后，我不知道会不会相同类别的图会相邻在一起，所以我将所有数据集shuffle 了一下。

3.2 Data augment

在进行了几次试验后我发现了一个问题：数据分布非常不平衡。于是我将TP、FP、FN、TN都打印了出来，如Figure 1所示， 有的类出现了一千多次但有的类仅出现了不到十次，而且大多数预测错误的情况都是将标签为1 的标签预测为0。由于小类的训练个数不足且大多数情况下小类的标签为0，这就导致

```
subset acc:  0.17178473700212832
[748, 37, 489, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[446, 39, 175, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[441, 407, 628, 130, 262, 28, 86, 58, 357, 481, 159, 74, 239, 197, 82, 31, 118, 74, 57, 65, 18, 32, 34, 14, 21, 26, 53, 1, 6]
[1654, 2806, 1997, 3159, 3027, 3261, 3203, 3231, 2932, 2806, 3130, 3215, 3050, 3092, 3207, 3258, 3171, 3215, 3232, 3224, 3271, 3257, 3255, 3275, 3268, 3263, 3236, 3288, 3283]
```

Figure 1: TP、FP、FN、TN

了大类的预测效果比较好而小类的预测效果非常差。于是我想进行数据增广用来增加训练集的数量，以便达到更好的效果。
这部分代码见111.py。

3.2.1 Data augment method 1

首先我使用了keras.preprocessing.image.ImageDataGenerator进行数据增广，利用旋转图片的方式得到新的训练样本。第一种方法中，我将所有的数据都增广了若干倍，这样之后的效果稍稍比之前好一点。但是，这样增广之后问题依然存在，就是那些小类的样本依然很小，比如那个最小的类本来就只有9张图，在我增强了4倍之后他依然只有36张图，依然很小。也就是说之前提到的问题依然没有解决。
这部分代码见Generation.py。

3.2.2 Data augment method 2

这里我依然使用keras.preprocessing.image.ImageDataGenerator进行数据增广，利用旋转图片的方式得到新的训练样本。在这里，我先罗列出我想要增广的类，把我想要增广的类的序号列在一个list 中，然后将包含有这些序号的样本挑出来，然后对挑出来的这些样本进行增广，这样我就只对小类进行增广。最后的结果是subset accuracy差不多而f1 score明显提高。
这部分代码见Generation2.py。

3.2.3 Data augment method 3

这里我发现了前两种方法的一个问题，就是在旋转之后的图片颜色改变了。如Figure 2 所示， 我也不知

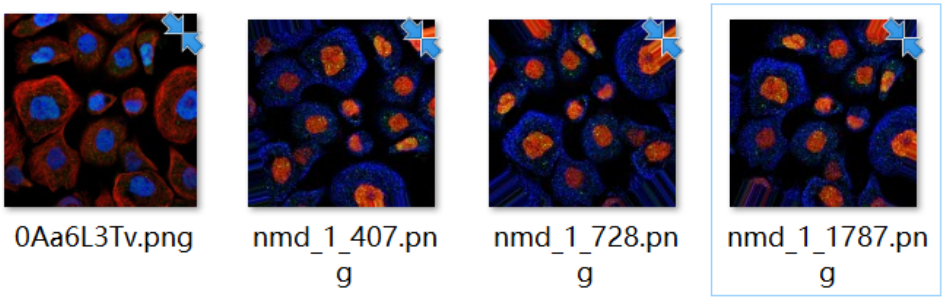

Figure 2: new sample

道为什么，可能是这个库的问题，我觉得这种变化很大的影响了训练数据的质量，因为我觉得同一种细胞的颜色应该是一样的。一方面，我使用了另外的保存方式是保存下来的图片颜色不变，另一方面我使用了另一种数据增强的方法：mix-up。即把两张图放在一起加一加除以二然后标签取并集。和上一种方法一样我先罗列出我想要增广的类，把我想要增广的类的序号列在一个list 中，然后将包含有这些序号的样本挑出来，然后对挑出来的这些样本进行增广，这样我就只对小类进行增广。我想有一个好处，这样能使标签集变得稍微不那么稀疏，因为标签为1 的比例增加了。
这部分代码见generation3.py。

3.3 Define loss function

由于数据分布非常不平衡，小类的数量很少，所以小类的预测结果非常差，这就导致了小类的f1很差，从而影响了整个任务的f1，于是我重新定义了自己的损失函数，手动定义了各个类的权重，我把小类的权重调整得比较大，这样训练时它就会对小类比较重视。最后这样的结果是subset小幅度下降，因为大类的权重低了所以对大类的预测稍微差了一点，但f1 score大幅度提高，因为小类的f1提高了。
这部分代码见222.py中的ourloss()函数。

3.4 CNN

我搭建了一个简单的CNN来测试效果，我加了两个卷积层和池化层，然后在最后加了两个全连接层用来分类，损失函数我选择使用了二元交叉熵函数，相当于做了29个二分类问题，分别判断是否属于一个类。
这部分代码见222.py。

3.5 Train ResNet50

接着我用在keras上重新训练了一个resnet50网络，由于resnet50对224*224的效果最好，所以我把用resize成224*224的图片作为网络的输入。
这部分代码见resnet50v6.py。

4 Part of Yingzhe Luo

4.1 Key problem of this Project

我认为该任务的主要难点大致有三处：一是样本的不均衡性，最大样本存在几千例而最小样本只有不到十例，这可以通过数据增广与自定义损失函数来进行处理，但由于是多标签任务，对小类增广时也会导致大类样本数更多，所以数据增广的效果也有一定局限性；二是复杂网络模型训不动的问题，这可以通过GPU和服务器训练等方法处理；三是从模型输出结果到最终预测结果的阈值不清楚，目前只能通过手动调试，过拟合风险较大。

4.2 Loss function design idea

起初在设计损失函数时，我们认为交叉熵损失函数可能会存在过于自信的问题，这是由于softmax损失函数自身的局限性带来的。比如对于真实值为$(1,0,0)$的标签来说，其输出值为(z_1,z_2,z_3), 那么softmax之后就是$\frac{e_1^z}{e_1^z+e_2^z+e_3^z}$，当将交叉熵作为损失函数时，该softmax之前的模型就可以发现，等比例增加z_1,z_2,z_3的值可以达到降低Loss的效果，这就会导致过于自信的问题。不过，对于我们的分类器来说，因为采用的是sigmoid，实质上是看作29个二分类问题，所以不会存在这个softmax过于自信的问题。根据这种思想，我们认为，可以考虑构造一种加权的交叉熵损失函数，来起到平衡大类小类样本的作用。

4.3 ResNet

残差网络解决了增加深度带来的副作用，退化问题，更容易优化，15年出世便斩获图像分类、检测、定位三项冠军。残差网络是通过浅层网络同等映射深层网络，深层网络并没有比浅层网络表现更佳，因为深层网络优化较难，并且存在局部解的问题，原因可能是深层网络并没有那么好求解。其核心思想是直接让$H(x)=x$ 训练会比较困难，但如果让$F(x)=H(x)-x=0$就会比较容易,去掉相同的主体部分，从而突出微弱变化的影响。我用预训练好的的resnet50模型对细胞图片进行了特征提取，最后通过全连接层与sigmoid 层得到输出，自定义阈值之后得到预测结果。

5 Part of Yufan Cai

5.1 Preprocess

如同前文黄晟原同学所说，图片数据集存在图片大小不一，图片数量较大的问题。首先，对于图片大小不一，有两种处理方式，第一种是简单地将所有图片resize为同一大小，但这种resize很难保存原始图片的所有信息，并且由于电脑的内存限制，无法resize成较大的尺寸。经过尝试，我发现使用120G内存的服务器能够最大将图片resize为512*512像素的大小。同时，在后续的实验中，我们发现在图片尺寸大于256*256像素的时候，模型的性能提升空间并不大。所以使用合适的图片大小能够加大训练速度的同时，不会显著降低模型的性能。

其次，第二种方法是使用Spatial Pyramid Pooling(Figure 3)。在一般的CNN结构中，在卷积层后面通常连接着全连接。而全连接层的特征数是固定的，所以在网络输入的时候，会固定输入的大小(fixed-size)。但在现实中，我们的输入的图像尺寸总是不能满足输入时要求的大小。然而通常的手法就是裁

剪(crop)和拉伸(warp)。这样的做法的缺陷在于：图像的纵横比和输入图像的尺寸是被改变的。这样就会扭曲原始的图像。而Kaiming He在这里提出了一个SPP(Spatial Pyramid Pooling)层能很好的解决这样的问题，这个方法的特点在于：1) 不管输入尺寸是怎样，SPP 可以产生固定大小的输出。2) 使用多个窗口(pooling window)。3) SPP 可以使用同一图像不同尺寸(scale)作为输入, 得到同样长度的池化特征。通过这个方法，能够达到这样的目的：1) 多窗口的Pooling会提高实验的准确率。2) 输入同一图像的不同尺寸，会提高实验准确率(从尺度空间来看，提高了尺度不变性(scale invariance))。3) 用了Multi-view来测试，也提高了测试结果。4)图像输入的尺寸对实验的结果是有影响的(因为目标特征区域有大有有小)。5)因为我们替代的是网络的Poooling层，对整个网络结构没有影响，所以可以使得整个网络可以正常训练。

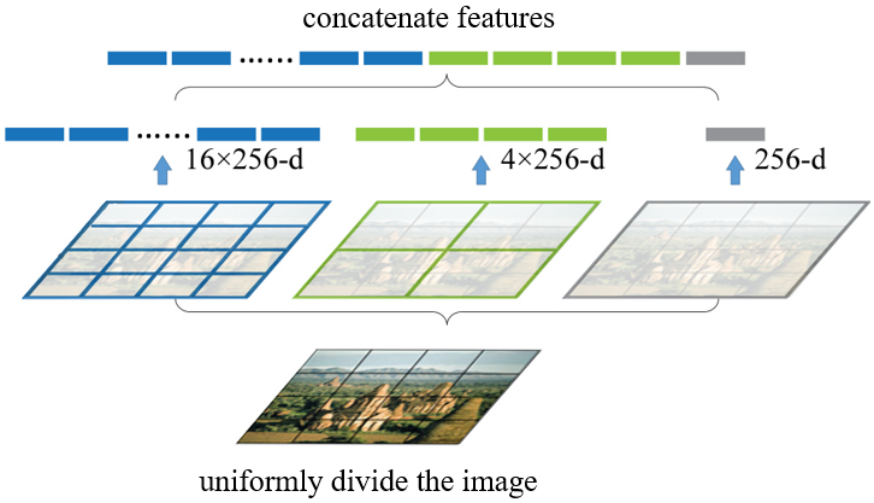

Figure 3: Spatial Pyramid Pooling

对于图片数量较大的问题，我首先尝试使用了华为云，但由于资费较高，传输速度较慢（已经将老师提供的华为云优惠券金额用完），我最后还是使用了实验室的服务器进行数据预处理，但由于服务器的资源限制（内存为128G），只能选择256*256的数据增广后的训练集，以及512*512像素大小的未进行数据增广的训练集。

5.2 Data analysis

数据的主要的问题是正负样本不均衡。如Figure 4所示，Nucleoplasm与Cytosol 多达三千多个，而Aggresome只有八个。所以我们考虑使用数据增广，以扩大小类的样本数目。如Figure 5所示，该结果是使用了数据增广之后的样本个数，可以发现，在进行数据增广之后，样本不均衡的程度并没有得到很大的改善，其中最大的两个类出现的数目都超过了5000，而最小的类才出现了85 次，这导致大类的预测效果明显优于小类。虽然数据增广能够增大小类的个数，但小类往往与大类出现在同一张图片中，这导致增广小类的同时，也同时增广了大类，导致数据增广有一定的效果，但没有预期的显著效果。

5.2.1 Data augment method and define loss function

数据增广是深度学习中常用的技巧之一，主要用于增加训练数据集，让数据集尽可能的多样化，使得训练的模型具有更强的泛化能力。当训练机器学习模型的时候，实际上实在调整它的参数，使得可以跟一个特定的输入符合。优化的目标是chase that sweet spot where our model’s loss is low。当前最好的神经网络拥有的参数量是上百万的量级。因此，有这么多的参数，就需要a proportional amount of examples 来学习这些参数。此外，通过数据增广提升数据集中的相关数据，能防止网络学习到不相关的特征，更多的学到更数据有关的性能，显著的提升整体的性能。

图像的数据增广方式包括很多。从几何角度来看，常用的有：水平翻转，一定程度的位移，裁剪，旋转等。从像素变换来看，常用的有：颜色抖动，增加噪声，例如高斯噪声等。此外还可以尝试多种操作的组合，例如同时做旋转和随机尺度变换。

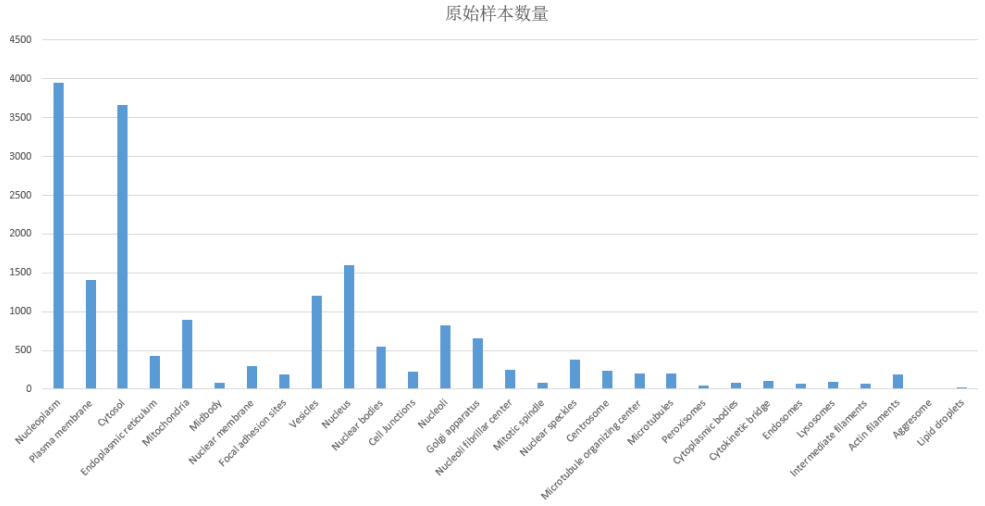

Figure 4: Sample amount

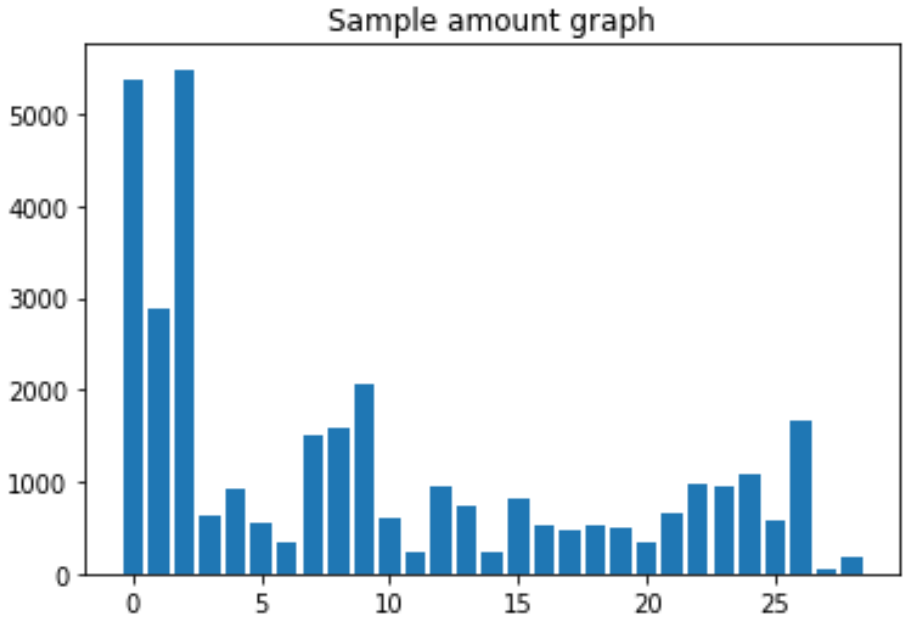

Figure 5: Sample amount

在本次实验中，我将所有包含小类的图片都通过旋转翻转等方法进行数据增广，从而扩大小样本的数量，但正如上文所说，这样得到的样本仍存在不同类数量差距大，正负样本均衡的问题，因此，为了让小类能够被更好地识别出来，我使用了自定义的损失函数，从而增大对小类不正确预测的惩罚，以及小类正确预测的奖励，使得模型性能有一定的提升。

5.3 VGG16

不同于同组的两位同学，我使用的是VGG16的模型。首先是不使用数据增广的结果，模型输入是512*512维大小的向量。

其次是使用了数据增广的结果，模型输入是128*128维大小的向量。可以看出，使用了数据增广的模型性能有较大的提升，其中宏平均F1-score有将近25%的提升,但是subset accuracy的提升较少。分析结果的原因，主要是因为在进行数据增广之后的样本分布仍不均衡，所以模型的预测仍不够好。

最后是使用了数据增广以及自定义了损失函数的方法，模型输入是128*128维大小的向量。从结果中可以看出，改变损失函数对性能的提升有一定的效果，但并不显著，同时会大大加大训练的轮数。我画出了Precision和recall的图像，如Figure 8和Figure 9。可以明显看出，对于数目较大类，例如第0、1、2类，precision在60%左右而recall 在0.55%左右，而对于小类，则是precision很高，而recall较低，这说明

是否使用数据增广	训练轮数	训练集		测试集	
		F1-score	accuracy	F1-score	accuracy
不使用数据增广，512维*512维输入	20	0.0496	0.2080	0.0438	0.1720
	40	0.1056	0.2580	0.0555	0.2010
	60	0.1411	0.3026	0.0616	0.2046
	80	0.2008	0.3717	0.0740	0.2064
	100	0.3227	0.4687	0.0949	0.2057
	150	0.4713	0.6211	0.1250	0.2084
	200	0.5907	0.7234	0.1407	0.2117
	250	0.6850	0.8009	0.1593	0.2107
	300	0.7830	0.8524	0.1652	0.2082
使用数据增广，128*128维的输入，不使用自定义loss	20			0.3628	0.2340
	40				
	60	0.9605	0.9029	0.4030	0.2561
	80	0.9887	0.9657	0.3958	0.2565
	100	0.9967	0.9891	0.3932	0.2548
使用数据增广，128*128维的输入，使用自定义loss	20			0.1824	0.1437
	40			0.3534	0.2147
	60			0.3839	0.2423
	80			0.3921	0.2417
	100			0.4042	0.2461
	120			0.4047	0.2491
	140	0.9966	0.9742	0.4029	0.2561
	160			0.4043	0.2574
	180			0.4028	0.2586
	200			0.4083	0.2616
	220	0.9997	0.9962	0.4061	0.2580

Figure 6: VGG16 result

模型还是存在对小类预测性能较差的问题。由于修改损失函数的权重存在很大的随机性，目前还没有获得更好的结果，但从目前的情况来看，修改损失函数，调大对于正确预测小类的奖励，能够增大模型的性能。

5.4 Partition and Hierarchical method

经过上面的分析，我首先尝试了将大类和小类分开预测的方法，这种方法能使得训练时期的正负样本较为均衡，从结果中可以看出，对于训练大类的模型，由于样本数量足够多，正负样本较为均衡，模型的训练效果较好，在较少轮的训练后，就已经接近收敛。而对于小类，虽然subset accuracy较之前的模型有了较大的提升，但对于宏平均F1-score并没有提高。最后将两个预测结果结合起来，发现最终的宏平均F1-score 为0.3990，而最终的subset accuracy 是0.256。这个结果与上面使用了数据增广后的模型性能相似，但是并没有提升。

为了获得更好的效果，我阅读了一些图像处理的论文，发现了一种层次分类的方法。首先查找文献，对原始数据label进行层次分类，分类的依据来自于网站https://www.proteinatlas.org/humanproteome/cell。经过三层的层次分类，我发现最高层次有三种，分别为Nucleus有7837种，Secretory有4163种，Cytoplasm有6028种，也就是说，我们可以首先进行一个三分类，每张图片可能属于某个类或者多个类，由于这样的层次划分，可以让属于同一个最高层次的图片同属于正样本或者负样本。也就是说，属于同一个最高类的样本有一些共同特征能够被模型学习，同样，属于同一个中间类的样本的一些公共特征能够被模型学习。由于之前做过层次分类的关系提取的模型，我们设想使用RNN，将每次得到的类作为隐藏变量向后传播。具体的方法还在测试之中。

对于何种类	训练轮数	训练集		测试集	
		F1-score	accuracy	F1-score	accuracy
大类预测	20			0.6140	0.5192
	40	0.9730	0.9591	0.6204	0.5518
	60	0.9962	0.9939	0.6229	0.5612
	80				
	100				
小类预测	20			0.1543	0.2970
	40			0.2687	0.3648
	60	0.7192	0.6652	0.2996	0.3483
	80	0.8749	0.8312	0.3376	0.3731
	100	0.9541	0.9366	0.3605	0.3831
	120	0.9861	0.9795	0.3724	0.3956
	140			0.3732	0.3952

Figure 7: VGG16 result2

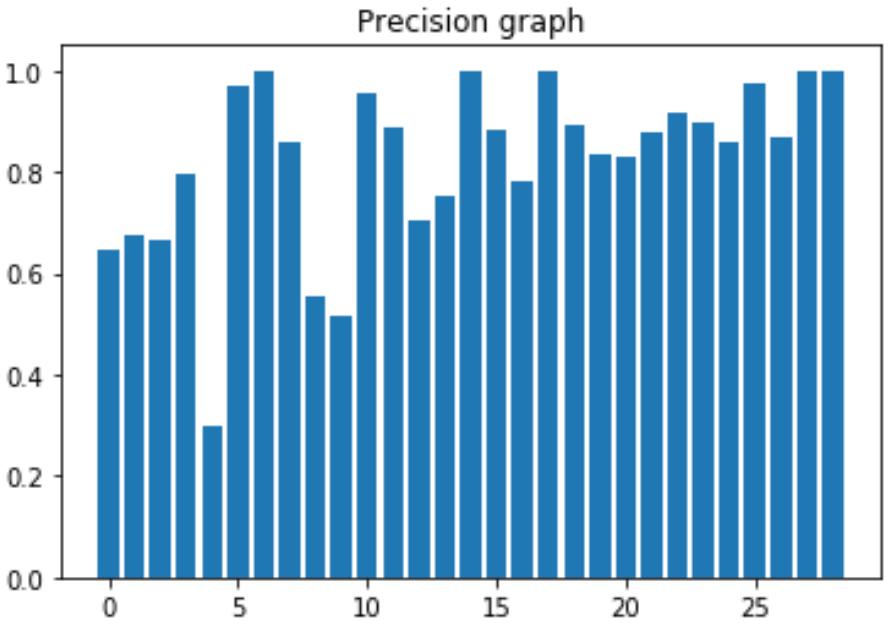

Figure 8: precision

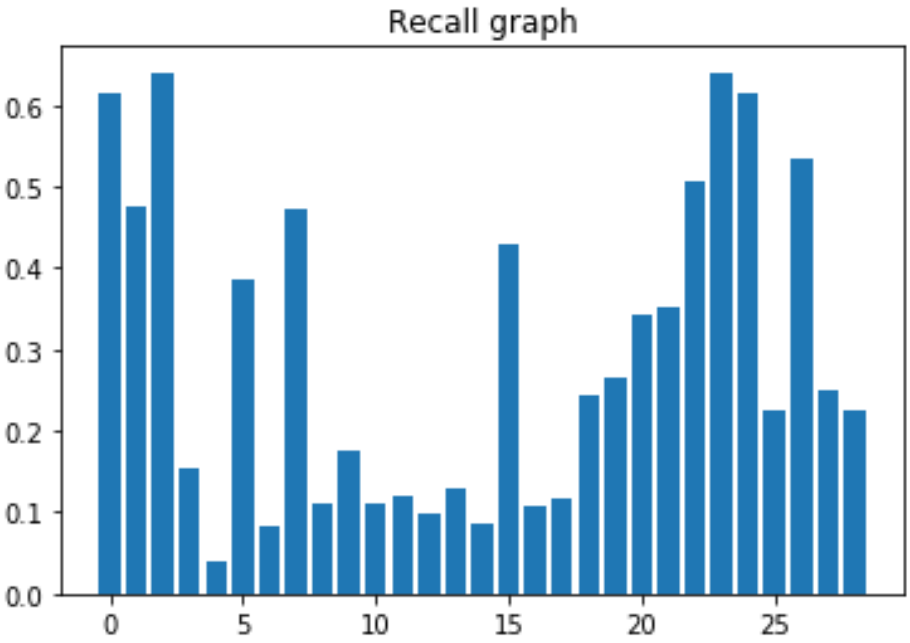

Figure 9: recall